Opinnäytetyö (AMK)

Kone- ja tuotantotekniikka

Koneautomaatio

2016

Otto Kurkijärvi

VIRTUAALISTEN ROBOTTISOLUJEN KEHITYS

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Kone- ja tuotantotekniikka | Koneautomaatio

2016 | 46

Sakari Koivunen

Otto Kurkijärvi

VIRTUAALISTEN ROBOTTISOLUJEN KEHITYS

Tämä insinöörityö käsittelee teollisuusrobotiikkaa, robottisolujen simulointia sekä etäohjelmointia. Työn tavoitteena oli luoda kolme virtuaalista robottisolua, joita on mahdollista käyttää robotiikan opetuksessa perehdyttämään opiskelijat ABB:n robotteihin, robottisolujen simulointiin sekä etäohjelmointimahdollisuuksiin. Toissijaisena tavoitteena oli suunnitella soluilla tehtäviä harjoitustöitä. Robottien vähäinen määrä sekä ABB:n valmistamien robottien puute Turun ammattikorkeakoulun robotiikan laboratoriossa olivat pääasialliset syyt tämän työn tekemiseen.

Robottisolujen luomiseen käytettiin RobotStudio-ohjelmistoa ja tarvittavien 3D-mallien luomiseen käytettiin Solidworks-ohjelmistoa. Työ tehtiin pääosin Turun ammattikorkeakoulun koneautomaatiolaboratorion tiloissa.

Työn tuloksena saatiin kolme toisistaan eroavaa robottisolusimulaatiota. Vaikka kaikki kolme solua ovat kappaleenkäsittelyä eroavat ne toisistaan muun muassa käsiteltävien kappaleiden, solujen monimutkaisuuden ja erilaisten ohjelmointitapojen ansiosta. Luodut simulaatiot antavat lisää mahdollisuuksia robotiikan käytännön opiskeluun Turun ammattikorkeakoulun koneautomaatio-opiskelijoille. Työstä saatiin myös tuloksena ohjeita RobotStudio-ohjelmiston käyttöön ja robottisolujen simulointiin.

ASIASANAT:

Simulointi, etäohjelmointi, RobotStudio

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Mechanical and Production engineering | Machine automation

2016 | 46

Sakari Koivunen

Otto Kurkijärvi

DEVELOPMENT OF VIRTUAL ROBOT CELLS

This thesis covers industrial robotics, simulation of industrial robot cells and offline-programming of industrial robots. Primary goal of this thesis was to create three different virtual robot cells. The cells are to be used in the education of students in the field of robotics with ABB robots, simulation of industrial robot cells and programming of industrial robots. Secondary goal was to design practical work to be done with the cells. Insufficient amount of robots and a lack of ABB robots in the robotics lab of Turku University of Applied Sciences, were the main reasons for this thesis.

RobotStudio computer program was used in the creation of the virtual robot cells. Solidworks program was used in the creation of the 3D-models used. The thesis was made in the machine automation lab of Turku University of Applied Sciences.

As a result of the thesis, three varying virtual robot cells and instructions on creating working simulations with RobotStudio, were created. Variance on the robot cells was achieved with different working objects and varying programming methods. The resulting virtual robot cells offer more chances at practical work on the field of robotics for machine automation students of Turku University of Applied Sciences in the future.

KEYWORDS:

Simulation, offline-programming, RobotStudio

SISÄLTÖ

KÄYTETYT LYHENTEET **6**

1 JOHDANTO **7**

2 TEOLLISUUSROBOTTI **8**

2.1 Robottien sovellusalat 8

2.2 Robottien rakenteet 9

2.2.1 Robottityypit 9

2.2.2 Robotin mekaniikka 11

2.3 Ohjausjärjestelmä 12

2.4 Teollisuusrobotin koordinaatistot 13

2.5 Työkalut 14

2.6 Robotin ohjelmointi 15

2.7 Teollisuusrobotiikan kehitysnäkymät 17

3 OHEISKOMPONENTTIEN LUONTI **18**

3.1 ABB RobotStudio 18

3.2 Solujen alustava suunnittelu 18

3.3 Oheislaitteiden ja työkalujen luonti 18

3.3.1 Työkalu 20

3.3.2 Kuljetin 25

3.3.3 Kuormalava 27

4 SOLUN ASETUKSET **31**

4.1 Solun layout ja robotin ohjaimen lisääminen 31

4.2 I/O-signaalit ja komponenttien linkitys 32

4.3 Liikeratojen luominen ja RAPID-muokkaus 34

5 LUODUT SIMULAATIOT **36**

5.1 Laatikkosolu 36

5.2 Sylinterisolu 38

5.3 Säkkisolu 42

6 POHDINTA **45**

KUVAT

Kuva 1. Robottityypit (ISO 8373). 10
Kuva 2. ABB IRB 2600 -kiertyvänivelinen teollisuusrobotti. 11
Kuva 3. ABB IRC5 -ohjausyksiköitä. 13
Kuva 4. Kaksisorminen vakiotarttuja (Schunk). 15
Kuva 5. ABB YuMi -robotti. 17
Kuva 6. Ohjelman aloitusikkuna. 19
Kuva 7. Home-välilehti. 20
Kuva 8. Modeling-välilehti. 21
Kuva 9. Create mechanism -ikkuna. 21
Kuva 10. Imukuppitarttujan komponentit. 22
Kuva 11. Imukuppitarttujan sidokset. 23
Kuva 12. Imukuppitarttujan I/O-signaalit. 24
Kuva 13. Imukuppitarttujan logiikkakaavio. 24
Kuva 14. Kuljettimen komponentit. 25
Kuva 15. Kuljettimen sidokset. 26
Kuva 16. Kuljettimen I/O-signaalit. 26
Kuva 17. Kuljettimen logiikkakaavio. 26
Kuva 18. Kuormalavan komponentit. 27
Kuva 19. Kuormalavan sidokset. 28
Kuva 20. Kuormalavan I/O-signaalit. 29
Kuva 21. Kuormalavan logiikkakaavio. 29
Kuva 22. Robotin ohjaimen asetusikkunat. 32
Kuva 23. Controller-välilehti. 32
Kuva 24. Uuden I/O-signaalin muokkausikkuna. 33
Kuva 25. Laatikkosoluun luodut I/O-signaalit. 33
Kuva 26. Simulation-välilehti. 34
Kuva 27. RAPID-välilehti. 35
Kuva 28. Tyhjä RAPID-editori. 35
Kuva 29. Laatikkosolu. 36
Kuva 30. Laatikkosolussa käytetty ohjelma. 37
Kuva 31. Sylinterisolu. 38
Kuva 32. Create conveyor- ikkunat. 40
Kuva 33. Sylinterin hionnassa käytetty ohjelma. 40
Kuva 34. Sylintereiden lavaukseen käytetty ohjelma. 41
Kuva 35. Säkkisolu. 42
Kuva 36. Säkkisolun pääohjelma. 43

KÄYTETYT LYHENTEET

TCP	Työkalupiste
IoT	Teollinen internet
SC	Smart Component, RobotStudiossa käytettävä älykomponentti

1 JOHDANTO

Fyysisten robottien pieni määrä, sekä ABB:n robottien puuttuminen Turun ammattikorkeakoulun robotiikan laboratoriossa olivat pääkohtaiset tarpeet tämän työn tekemiselle. Koska ABB on yksi suurimmista robottitoimittajista, olisi opiskelijat hyvä perehdyttää, ainakin jonkin asteisesti, myös ABB:n robotteihin.

Tämän työn tavoitteena oli suunnitella ja toteuttaa kolme virtuaalista teollisuusrobottisolua käytettäväksi Turun ammattikorkeakoulun koneautomaatiolinjan robotiikan opetuksessa. Luotuihin soluihin oli tarkoituksena myös suunnitella alustavasti joitakin harjoituksia, joilla opiskelijat voisivat harjoitella RobotStudio-ohjelmiston käyttöä.

Työn alussa luodaan katsaus teollisuusrobotiikkaan ja robotteihin yleisesti. Sen jälkeen kerrotaan komponenttien mallintamisesta ja luomisesta RobotStudio-ohjelmistolla. Seuraavaksi käsitellään solujen luomista ohjelmistolla. Lopuksi katsastetaan luotuja soluja ja pohditaan lopputulosta.

2 TEOLLISUUSROBOTTI

Teollisuusrobotin voidaan ajatella syntyneen 1960-luvulla, jolloin Unimation Company Inc. toimitti General Motorsin tehtaalle ensimmäisen Unimate-robottinsa. Ensimmäisen mikrotietokonella ohjatun, täysin sähköisen teollisuusrobotin kehitti ruotsalainen ASEA (nykyisin osa ABB:ta) (Nocks 2008, 23).

Standardin- ISO 8373 määritelmän mukaan teollisuusrobotti on uudelleen ohjelmoitavissa oleva monipuolinen, vähintään kolminivelinen mekaaninen laite, joka on suunniteltu liikuttamaan kappaleita, osia, työkaluja tai erikoislaitteita ohjelmoitavin liikkein monenlaisten tehtävien suorittamiseksi teollisuuden sovelluksissa (Suomen automaatioseura, 2016). Teollisuusrobotti on siis mekaaninen laite, joka liikuttaa siihen kiinnitettyä työkalua haluttuun paikkaan halutussa asennossa.

2.1 Robottien sovellusalat

Vuonna 2014 yleisimmät teollisuudenalat, joilla robotteja käytettiin, olivat auto-, elektroniikka-, metalli-, muovi- ja elintarviketeollisuus (IFR 2015).

Tyypillisiä sovelluskohteita roboteille ovat (Keinänen ym. 2007, 259)

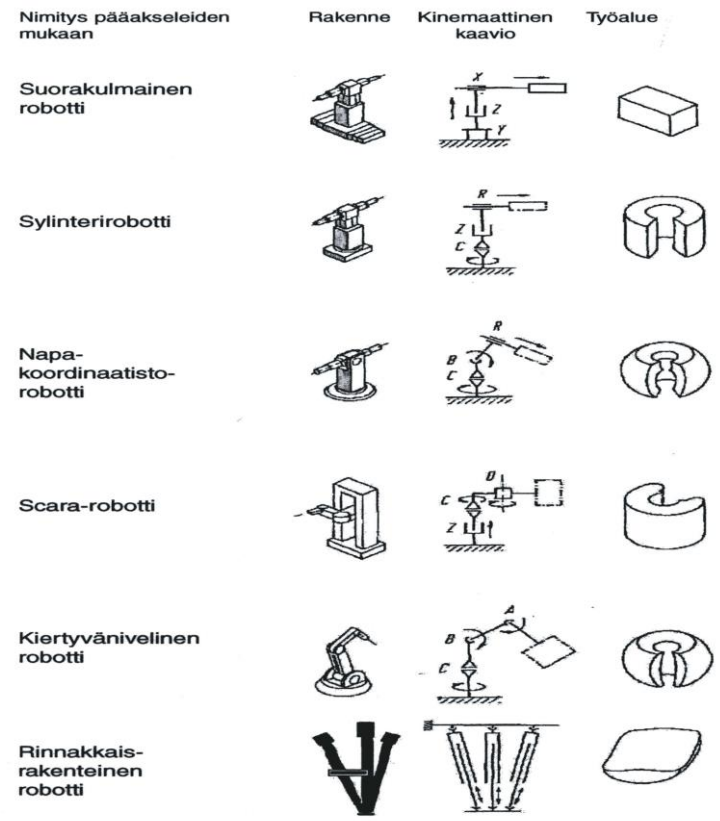
- kokoonpano
- hitsaus
- kappaleenkäsittely
- konepalvelu
- pakkaus.

2.2 Robottien rakenteet

2.2.1 Robottityypit

Robotit jaetaan eri tyyppeihin niiden mekaanisten rakenteiden mukaan. Eri robottityypeillä on useimmiten eri määrä vapausasteita, mikä vaikuttaa niiden työskentelyalueeseen. Mitä enemmän vapausasteita robotissa on, sitä monimuotoisempi sen työskentelyalue on. Standardi- ISO 8373(kuva 1) jakaa robotit tyyppeihin seuraavasti (Sclater & Chironis 2006, 53 - 65):

- suorakulmaiset robotit
- sylinterirobotit
- napakoordinaatistorobotit
- SCARA-robotit
- kiertyväniveliset robotit
- rinnakkaisrakenteiset robotit



Kuva 1. Robottityypit (ISO 8373).

Näistä vakiintuneimpia robottityyppejä ovat suorakulmaiset robotit, SCARA- robotit sekä kiertyväniveliset robotit. Yleisin robottityyppi teollisuudessa on kiertyvänivelinen robotti (kuva 2) sen monipuolisesta työskentelyalueesta johtuvan laajan sovellusalan takia. (Aalto ym. 1999, 16 - 18).

Kuva 2. ABB IRB 2600 -kiertyvänivelinen teollisuusrobotti.

2.2.2 Robotin mekaniikka

Kuten aiemmin mainittiin, robotti on mekaaninen laite, joka liikuttaa siihen kytkettyä työkalua halutulla tavalla. Robotin mekaaninen osa, esimerkiksi nivelvarsirobotin käsivarsi, koostuu tukivarsista ja nivelistä. Niveliä liikuttavat toimilaitteet ovat nykypäivänä useimmiten sähköisiä servomoottoreita, suuremmilla kuormilla käytetään myös hydraulisia toimilaitteita. Venttiiliteknologian parantuessa ovat myös pneumaattiset toimilaitteet yleistyneet. Voiman välitykseen roboteissa käytetään useimmiten hammaspyörästöjä. Muita välitysmenetelmiä ovat työntötangot, kuularuuvit, ketjut ja hammashihnat. Muita roboteista yleisesti löytyviä mekaanisia osia ovat alennusvaihteet sekä jarrut (Aalto ym. 1999, 19 - 20).

2.3 Ohjausjärjestelmä

Mekaanista konetta tärkeämpi robottijärjestelmän osa on ohjausjärjestelmä, jonka verrattainen työpanos on suurempi, ja joka on myös teknisesti vaikeampi toteuttaa kuin robotin mekaniikka (Aalto ym. 1999, 15).

Robottijärjestelmän komponentit ovat

- työkalu
- prosessianturit/ -elimet
- mekaaninen manipulaattori
- ohjausjärjestelmä
- mahdolliset oheislaitteet
- liitännät ulkoisiin tietokoneisiin.

Robottien ohjausjärjestelmät (kuva 3) ovat reaaliaikaisia prosessitietokoneita, jotka voivat ohjata robotin toimilaitteita tuhansia kertoja sekunnissa ja pystyvät reagoimaan anturiviesteihin millisekunneissa (Aalto ym. 1999, 34).

Ohjausjärjestelmän yleisesti sisältämät laitteet ovat

- keskusyksikkö
- massamuisti
- käsiohjain, ohjelmointiin
- liitännät ulkoisille tietokoneille
- nivelkohtaiset servotoimilaitteet
- teholähteet, jotka muuntavat sähkötehon laitteille sopivaksi.

Robotin ohjausjärjestelmän tehtävät voidaan jakaa viiteen eri luokkaan (Aalto ym. 1999, 35)

- liikeohjauksen tehtävät
- ohjelmointiin liittyvät tehtävät
- ohjelmien toisto
- turvallisuustoiminnot

- huolto- ja käyttöönottotehtävät.

Kuva 3. ABB IRC5 -ohjausyksiköitä.

2.4 Teollisuusrobotin koordinaatistot

Standardi- ISO 9787-1990 määrittelee teollisuusroboteille koordinaatistot seuraavasti

- Maailmankoordinaatisto on robotin ympäristöön esimerkiks rakennukseen tai kuljettimeen sidottu robotin ulkopuolinen koordinaatisto.
- Peruskoordinaatisto on sidottu robotin jalustaan. Tavallisesti Z-akseli yhtyy ensimmäiseen vapausasteen akseliin, X-akseli osoittaa ensimmäisen nivelen työalueen keskikohtaan ja XY-taso yhtyy lattiaan.

- Työkalukoordinaatisto on suorakulmainen ja sidotaan työkalumäärityksellä haluttuun kohtaan robotin työkalua (Keinänen ym. 2007, 260 - 261).

Näiden lisäksi voidaan määrittää käyttäjän koordinaatisto, joka voidaan sitoa mihin tahansa kohtaan ja asentoon robotin työalueella.

2.5 Työkalut

Robottijärjestelmässä työkalulla tarkoitetaan sitä mekaanista osaa, jota robotti liikuttaa haluttuun paikkaan. Työkalu on robottijärjestelmän lähes tärkein komponentti (Aalto ym. 2009, 60).

Työkalut voidaan jakaa

- tarttujiin
- prosessiin osallistuviin työkaluihin
- erikoislaitteisiin, kuten kamerat ja laserskannerit.

Tavallisimpia esimerkkejä prosessiin osallistuvista työkaluista ovat hitsauspistoolit, ruiskumaalaus- ja liimaussuuttimet, työstävät työkalut, kuten jyrsimet ja hiomatyökalut. Edellä mainitut ovat vain esimerkkejä yleisimmistä prosessiin osallistuvista työkaluista ja niiden lisäksi on vielä useita muita erilaisiin prosessiin osallistuvia työkaluja (Aalto ym. 2009, 76 - 77).

Tarttujat

Tarttujan (kuva 4) suunnittelu ja/tai valinta on yksi tärkeimmistä vaiheista robottijärjestelmän suunnittelussa. Tarttujaa suunnitellessa usein pyritään saamaan tarttujasta rakenteeltaan mahdollisimman yksinkertainen, pienikokoinen ja kevyt. Tarttujaa valittaessa/suunniteltaessa tulisi tuntea tarttujien tyypit sekä tartuntatavat (Aalto ym. 1999, 60 - 65).

Tarttujat voidaan jakaa tartuntatapansa mukaan ryhmiin seuraavasti:

- mekaaniset tarttujat
- imu- ja tyhjiötarttujat

- magneettitarttujat
- erikoistarttujat
- vakiotarttujat.

Kuva 4. Kaksisorminen vakiotarttuja (Schunk).

Kuvassa 4. Schunk:n valmistama paineilmatoiminen kaksisorminen vakiotarttuja.

2.6 Robotin ohjelmointi

Robotin ohjelmoinnissa on useita huomioon otettavia seikkoja, mutta ohjelmoinnin päätehtävät voidaan jakaa seuraavasti:

- Toimintajärjestyksen ja logiikan luominen sovelluksessa tarvittaville robotin liikkeille
- Robotin synkronointi järjestelmässä olevien oheislaitteiden kanssa, sekä tiedon välitys laitteiden kesken
- Virhetilanteissa robotin toiminnan määrittely.

Robotin ohjelmoimiseen on kaksi menetelmää, jotka ovat online- sekä offline-ohjelmointi. Online-menetelmä sisältää johdattamalla sekä opettamalla ohjelmoinnin.

Johdattamalla ohjelmointi on yksinkertaisin menetelmä, siinä robotin nivelet vapautetaan ja liikutetaan manuaalisesti robottia haluttua liikerataa pitkin. Robotti kirjoittaa sen muistiinsa ja pystyy tämän jälkeen toistamaan opetetun liikeradan. Johdattamalla ohjelmoinnin huonoina puolina ovat ohjelman hankala muokattavuus sekä liikeratojen heikko tarkkuus.

Opettamalla ohjelmoinnissa robotin työkalu ohjataan haluttuun paikkaan ja asentoon käsiohjaimella ja tallennetaan paikka järjestelmän muistiin. Aseman lisäksi määritetään robotille haluttu liiketapa sekä -nopeus. Näitä pisteitä tallennetaan tyypillisesti muutamia, joista ohjausjärjestelmä laskee robotille liikeradan. Tätä menetelmää käytetään usein yhdessä tekstipohjaisen ohjelmoinnin kanssa, jolloin käsiohjaimella opetetaan robotille liikerata ja tekstieditorissa kirjoitetaan robotille I/O- käskyt ja muu toimintalogiikka.

Offline- eli etäohjelmoinnissa robotin liikerata ja toimintalogiikka muodostetaan ulkopuolisella tietokoneella ja ladataan sieltä robotin ohjausjärjestelmään. Mallipohjaisessa etäohjelmoinnissa robottisolusta luodaan tarkka 3D-mallinnos simulointiohjelmistoon, jossa simuloitu solu toimii samalla tavalla kuin oikeassakin solussa. Jokaisella robottitoimittajalla on oma etäohjelmointiohjelmistonsa ja oma ohjelmakieli, jota robotit lukevat, esimerkiksi tässä työssä käytetty ABB RobotStudio ja ABB-robottien RAPID-kieli. On myös olemassa ohjelmistoja, joilla voidaan ohjelmoida minkä tahansa robottivalmistajan robotteja. Tällöin robotin ohjelmat luodaan käyttäen yleiskieltä, joka on käännettävä käytetyn robotin käyttämälle kielelle, kun ohjelma ladataan robotin ohjaimeen. (Keinänen ym. 2007 262 - 263).

Mallipohjainen etäohjelmointi on menetelmistä yleisin ja tehokkain, sillä siinä ei tarvitse pysäyttää edellistä ohjelmaa ohjelmoinnin ajaksi ja ohjelma voidaan verifioida tarkasti etukäteen simuloinnin avulla (Aalto ym. 1999 82 - 86).

2.7 Teollisuusrobotiikan kehitysnäkymät

Kaksikäsivartiset, ihmisten keskuudessa työskentelyyn pystyvät robotit, esimerkiksi ABB YuMi (kuva 5), ovat teollisuusrobotiikan viimeaikainen kehitysaskel. Tämän lisäksi viime vuosina teollisuusrobottien teknologinen kehitys on ollut hidasta, eivätkä robottivalmistajat ennusta robottien kehittyvän erikoisemmin lähitulevaisuudessa.

Kuva 5. ABB YuMi -robotti.

Robottivalmistajat sen sijaan ennustavat sensoreiden, robottien työkalujen sekä robottien ohjelmistojen kehityksen olevan vilkasta tulevaisuudessa. Älykkäämmät sensorit ja paremmat ohjelmistot roboteissa mahdollistavat paremman IoT:n hyödyntämisen. Paremman IoT:n hyödyntämisen myötä koneet saadaan keräämään ja analysoimaan enemmän dataa ja tulkitsemaan paremmin ympäristöään, jolloin järjestelmistä saadaan itsenäisempiä sekä mahdollistetaan robotin ja ihmisen yhteistyö. Pehmeistä materiaaleista valmistetut tarttujat mahdollistavat tartunnan kappaleisiin, joihin se ei aikaisemmin ole ollut mahdollista. Nivelvarsirobottien käyttö 3D-tulostuksessa on myös robotiikan uusi suuntaus, joka todennäköisesti tulee yleistymään tulostinten sekä robottien kehittyessä (Anandan 2015).

3 OHEISKOMPONENTTIEN LUONTI

3.1 ABB RobotStudio

RobotStudio on ABB:n teollisuusrobottien etäohjelmointi- ja simulointiohjelmisto. Ohjelmisto perustuu ABB Virtual Controlleriin, joka on tarkka kopio oikeiden robottien ohjainten käyttämästä ohjelmistosta. Tästä johtuen voidaan ohjelmistolla luoda erittäin realistisia simulaatioita ja robottiohjelmia. Ohjelmistossa on myös tuottavuutta parantavia ominaisuuksia, joita voidaan käyttää koulutus-, ohjelmointi-, ja optimointitarkoituksissa häiritsemättä tuotantoa(abb.com 2016).

3.2 Solujen alustava suunnittelu

Toteutettaviin soluihin ei ollut fyysistä mallia tai niitä ei ollut sidottu minkään tilan, pohjapiirrokseen, joten layout-suunnittelu kulki pitkälti oman mielikuvituksen varassa, pyrkimyksenä kuitenkin, että solut muistuttaisivat todellisia tuotantosoluja. Simulaatioihin saatiin varianssia käyttämällä toisistaan poikkeavia työkappaleita, erilaisia oheislaitteita, erilaisia ohjelmointitapoja sekä erikokoisia robottimalleja. Työn aloituspalaverissa solujen työkappaleiksi määräytyivät säkki, laatikko sekä sylinterin mallinen kappale, jolloin tartunta kappaleisiin on mahdollisimman erilainen.

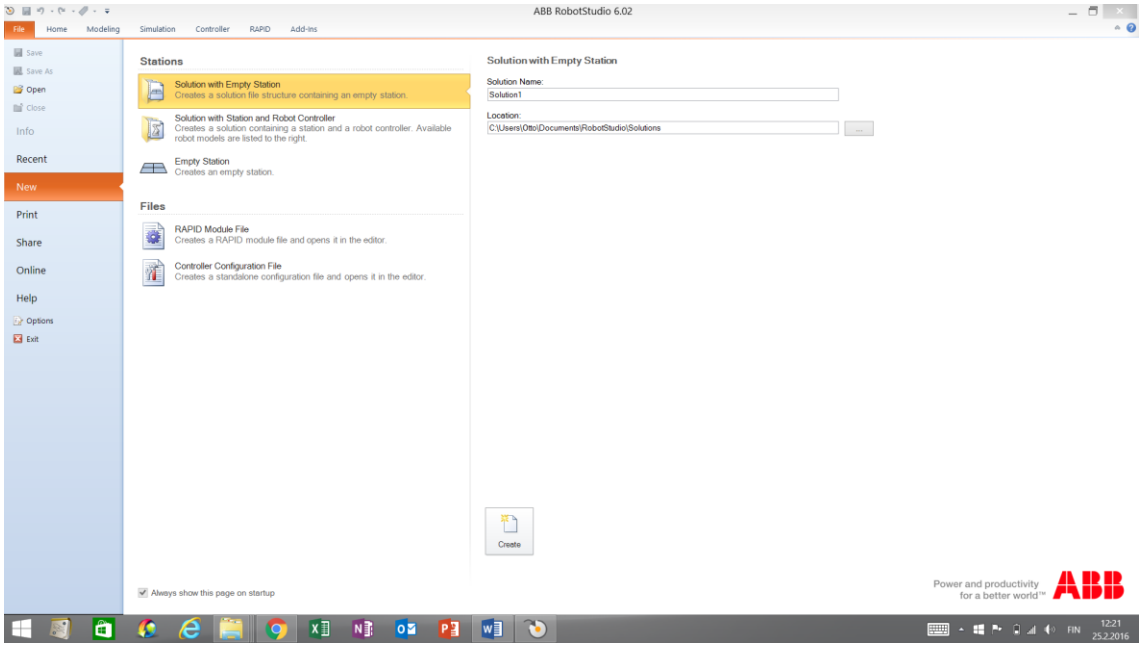
3.3 Oheislaitteiden ja työkalujen luonti

Vaikka solujen asettelu muuttui ja tarkentui projektin aikana, saatiin alkuperäisestä suunnitelmasta kuitenkin selville tarvittavat oheislaitteet ja työkalut, jotka tuli mallintaa ja tuoda ohjelmistoon.

RobotStudio- ohjelmistoon on mahdollista tuoda itse tehtyjä malleja Import Geometry- toiminnolla. Ohjelmisto tukee useita eri tiedostomuotoja, tässä työssä käytettiin ACIS (.sat) -muotoa, sillä se tuntui parhaiten toimivan tarkoitukseen.

Tuoduista geometrioista on mahdollista luoda älykkäitä komponentteja, Smart Component -toiminnon avulla, joilla voi välittää I/O-tietoa simulaatioissa, manipuloida muita kappaleita, esimerkiksi tarttujalla tarttua kappaleeseen ja liikuttaa sitä, tai vaikka esittää tekstiä tai ääntä simuloinnin aikana.

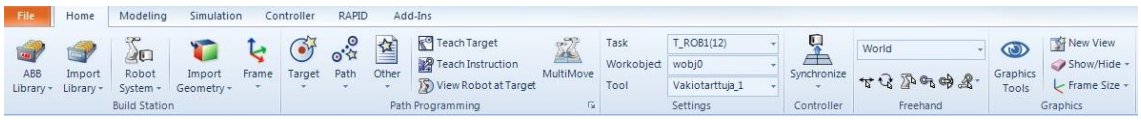
Komponenttien luonti on helpointa aloittaa tuomalla mallit tyhjään simulaatioeditoriin, johon ei ole lisätty robotin virtuaaliohjainta. Ohjelman aloitusikkunassa (kuva 6) valitaan Empty Staton -painike.



Kuva 6. Ohjelman aloitusikkuna.

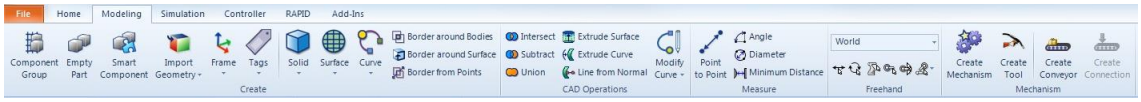
3.3.1 Työkalu

Työkalun luominen alkaa geometrian tuomisella ohjelmaan Import Geometry- toiminnon avulla. Seuraavaksi tuodun kappaleen origo siirretään keskelle kiinnityslaipan pohjaa samansuuntaisesti kuin robotin kiinnityslaipan origo, jotta työkalun liittäminen robottiin onnistuu halutulla tavalla. Tämä onnistuu käyttäen Set Local Origin -toimintoa, joka löytyy painamalla hiiren oikeaa painiketta kappaleen kohdalla ja valitsemalla Modify-> Set Local Origin. Yksinkertaisin tapa kappaleen origon asettamiseen on ensin siirtää tuotu kappale solun koordinaatiston nollapisteeseen Set Position -toiminnolla ja kääntää se Rotate-toiminnolla niin, että tuotu malli on kiinnityslaippa alaspäin ja kiinnityslaipan keskikohta on nollapisteessä. Tämän jälkeen voidaan Set Local Origin -ikkunaan valita kaikkiin kohtiin arvo 0, jolloin kappaleen origo on oikeassa paikassa. Seuraavaksi luodaan uusi koordinaattipiste, jota käytetään työkalun työkalupisteenä. Home-välilehdeltä (kuva 7) valitaan New Frame ja asetetaan se haluttuun pisteeseen, halutussa asennossa.



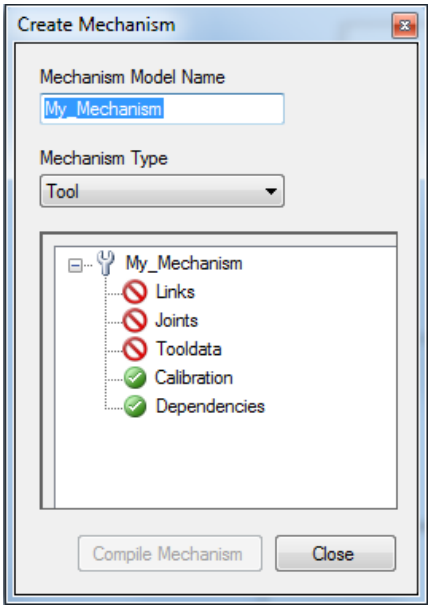
Kuva 7. Home-välilehti.

Seuraavaksi luodaan kappaleesta työkalu Modeling-välilehdeltä (kuva 8) löytyvän Create Tool -toiminnon avulla. Painiketta painettaessa avautuu ikkuna, jossa määritetään työkalulle nimi, sen geometria sekä massa, painopiste ja hitausmomentti. Massan, painopisteen tai hitausmomentin määrittäminen ei ole pakollista, mutta tekee simulaatiosta realistisemman. Määritysten jälkeen päästään Next-painiketta painamalla seuraavaan vaiheeseen, missä määritetään työkalupiste (TCP), sekä nimi kyseessä olevalle pisteelle. Kun pisteelle on annettu nimi, voidaan rastittaa valinta Values from Frame ja valita valikosta aiemmin luotu piste. Nyt voidaan painaa ikkunan keskellä olevaa nuolta, jolloin TCP-arvot siirtyvät oikeanpuoleiseen laatikkoon. Mikäli luodaan vain yksi työkalupiste, voidaan Done-painikkeella sulkea muokkausikkuna.



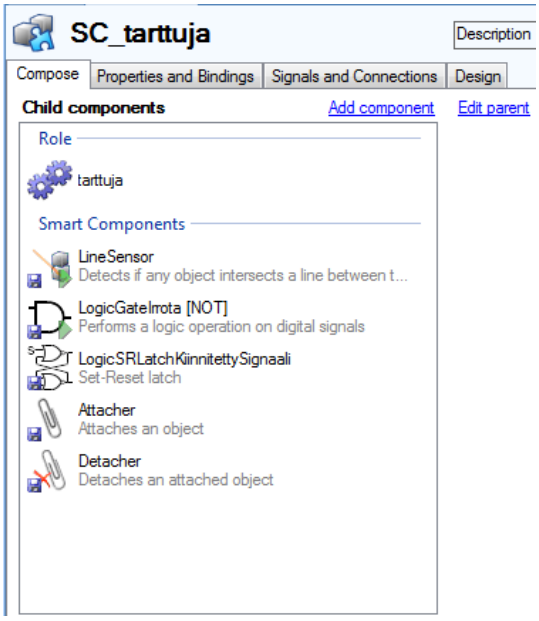
Kuva 8. Modeling-välilehti.

Mikäli työkalussa on tarkoitus olla liikkuvia osia, esimerkiksi tässä työssä käytetty sormitarttuja, on runko ja liikkuvat osat tuotava soluun erillisinä kappaleina. Kiinnityslaipan origo pitää asettaa samoin kuin aiemmin ja irralliset osat pitää asettaa paikoilleen. Kun irralliset osat on asetettu, pitää kappaleesta tehdä mekanismi Create Mechanism -toiminnolla, joka löytyy Modeling-välilehdeltä. Painiketta painettaessa oikeaan reunaan aukeaa ikkuna (kuva 9) missä määritetään mekanismin tyyppi, sen nivelten liitoskohdat, nivelten liiketyyppi ja -laajuus, työkalutiedot, mikäli mekanismi on työkalu ja liikkuvien osien asennot simulaatiossa ja niiden liikenopeus.



Kuva 9. Create mechanism -ikkuna.

Työkalun toiminnot luodaan Smart Component -toiminnolla, joka löytyy Modeli-välilehdeltä. Smart Component -painiketta painettaessa avautuu sen muokkausikkuna (kuva 10) ja SC-objekti ilmestyy vasemmalla olevaan puukaavioon. Seuraavaksi tuotu kappale raahataan SC-objektin päälle puukaaviossa, jolloin se linkittyy SC-komponentin käyttöön. Tuotu kappale ilmestyy muokkausruutuun, jolloin painetaan hiiren oikeaa painiketta objektin päällä ja valitaan Set as Role, jolloin aiemmin luodut työkalutiedot siirtyvät SC-komponentille. Add Component -painikkeella valitaan työkalulle halutut toiminnot.



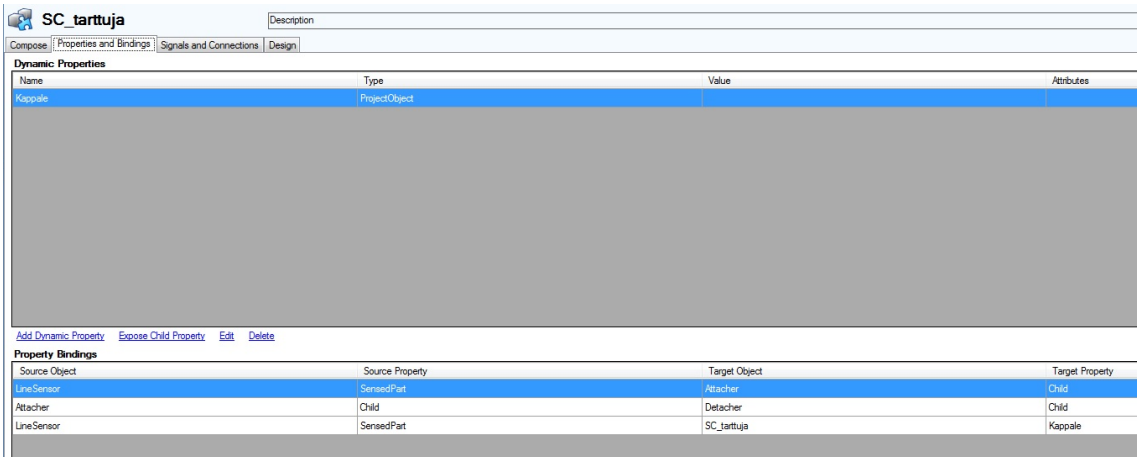
Kuva 10. Imukuppitarttujan komponentit.

Toiminnon lisäämisen jälkeen puukaavion yläpuolelle ilmestyy Properties-ikkuna, jossa voidaan muokata kyseisen komponentin asetuksia.

Imukuppitarttujan komponenttien asetukset:

- LineSensor-komponentti asetettiin oikeaan paikkaan ja määrättiin säteen halkaisija ja pituus
- LogicGate-komponentti asetettiin NOT-funktioksi
- Attacher-komponentille määrättiin Parent-asetukseen tarttujan grafiikka.

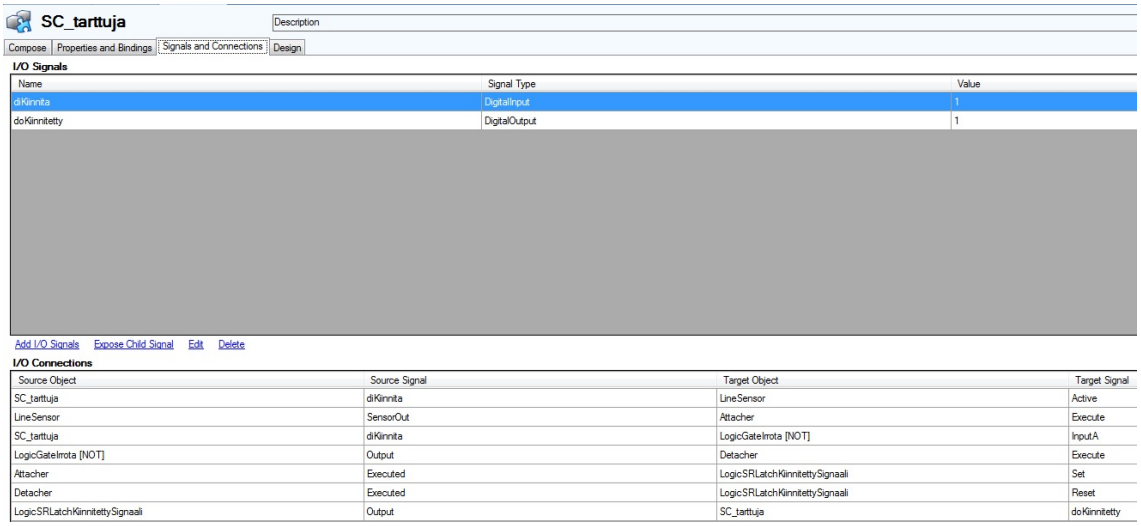
Properties and Bindings-välilehdellä (kuva 11) voidaan luoda sidoksia ja määrityksiä toimintojen välillä.



Kuva 11. Imukuppitarttujan sidokset.

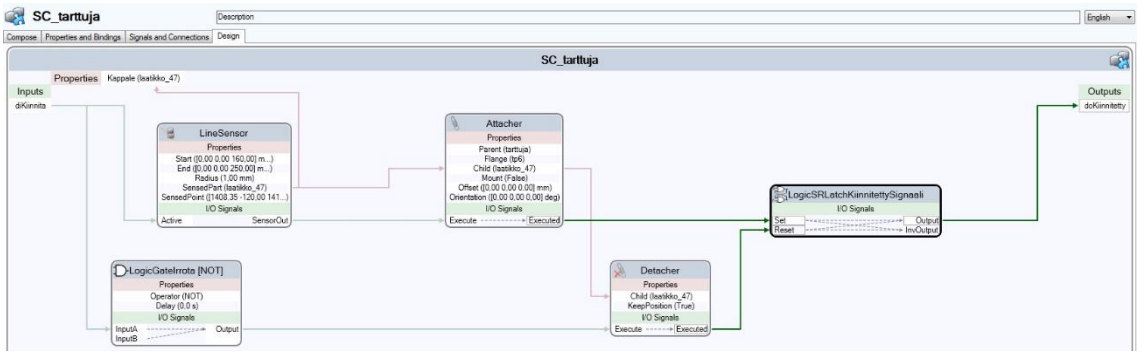
Sidosten toiminta on seuraava: LineSensor-komponentin tunnistaman kappaleen tunnistetiedot välitetään Attacher ja Detacher-komponenttien Child-asetukseen, jolloin ohjelmisto tietää mikä komponentti on työkalu ja mikä työkappale. Tarttujalle luotiin myös dynaaminen ominaisuus, Kappale, jolla voidaan tunnistetun kappaleen tiedot välittää muihin SC-komponentteihin.

Signals and Connections-välilehdellä (kuva 12) luodaan komponentille I/O-liitynnät sekä linkitetään lisättyjen toimintokomponenttien tulot ja lähdöt toisiinsa, jolloin luodaan laitteelle toimintalogiikka. Tarttujalle luotiin digitaalinen tulo, diKiinnita ja digitaalinen lähtö, doKiinnitetty.



Kuva 12. Imukuppitarttujan I/O-signaalit.

Design-välilehdellä (kuva 13) voidaan tarkastella komponentin toimintalogiikkaa blokkikaaviona. Luodut älykomponentit voidaan tallentaa klikkaamalla niitä vasemmalla olevassa puukaaviossa hiiren oikealla painikkeella ja valikosta valita Save as Library.



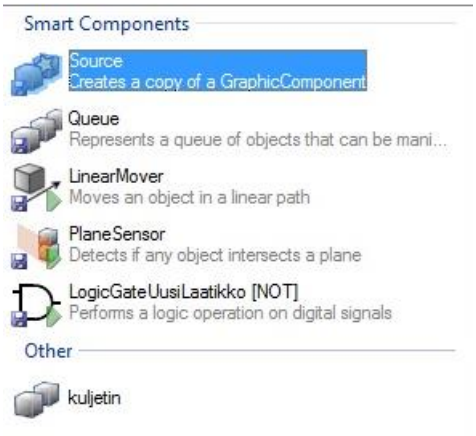
Kuva 13. Imukuppitarttujan logiikkakaavio.

Tarttujan logiikka toimii seuraavasti: kun tarttuja diKiinnita-tulo on aktiivinen, asettaa se LineSensor- komponentin aktiiviseksi ja LogicGate(NOT)-komponentin tulon InputA aktiiviseksi. LineSensor-komponentin tunnistettua kappaleen sen lähtö lähettää Attacher-komponentin tuloon signaalin ja tarttuja tarraa kappaleeseen. Tartunnan jälkeen Attacher-komponentin lähtö asettaa Set-Reset-komponentin Set-tulon aktiiviseksi ja komponentin lähtö antaa signaalin tarttujan lähtöön do-

Kiinnitetty signaalin, jolloin robotin ohjain tietää, että tartunta on onnistunut. di-Kiinnita-tulon ollessa ei aktiivinen, ei NOT-komponentin tulo ole aktiivinen ja De-tacher-komponentti irrottaa kappaleen tarttujasta ja asettaa Set-Reset-kom-ponentin lähdön ei aktiiviseksi, joka vuorostaan asettaa doKiinnitetty-lähdön ei aktiiviseksi. Set-Reset-komponentti siis simuloi alipainevahtia tarttujassa.

3.3.2 Kuljetin

Kuljetinmallin tuonti ohjelmistoon onnistuu samalla tavalla Import Geometry -toi-minnolla. Kuljettimen paikallista origoa ei tarvitse asettaa, mutta kappaleen liikut-telu on kätevämpää, mikäli origo on kuljettimen pohjassa. Kuljetin voidaan luoda käyttämällä Modeling-välilehdeltä löytyvää Create Conveyor -toimintoa, mutta täl-löin kuljettimeen ei voida lisätä toimintokomponentteja, antureita tms. Toinen tapa on luoda kuljettimesta Smart Component (kuva 14).



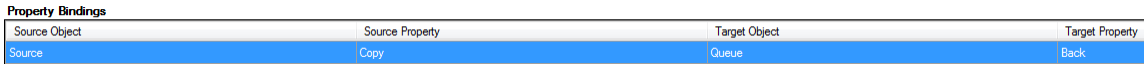
Kuva 14. Kuljettimen komponentit.

Kuljettimen komponentteihin tehdyt asetukset:

- Source-komponentin SourceCopy asetukseen valittiin työkappale.
- LinearMover-komponentille valittiin liikesuunta ja nopeus sekä sen liikut-tama objekti.
- PlaneSensor-komponentti asetettiin oikean kokoiseksi oikeaan kohtaan.

- LogicGate-komponentti valittiin NOT-funktioksi.

Source-komponentin ja Queue-komponentin välille luotiin sidos, jolloin Source-komponentin luoman kopion liittyessä jonokomponentin ryhmään liitetään se aina viimeiseksi jäseneksi (kuva 15).

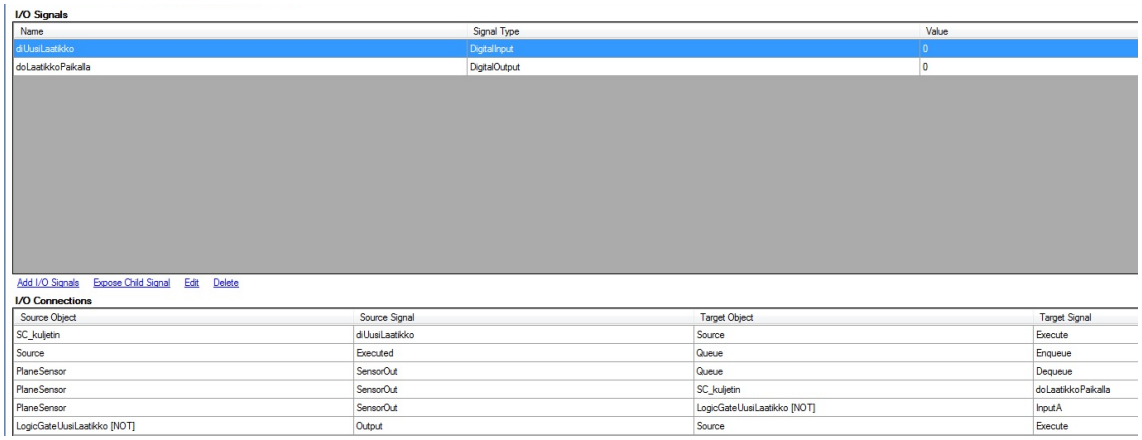


Property Bindings

Source Object	Source Property	Target Object	Target Property
Source	Copy	Queue	Back

Kuva 15. Kuljettimen sidokset.

Kuljettimelle luotiin yksi digitaalinen tulo, diUusiKappale ja yksi digitaalinen lähtö, doKappalePaikalla (kuva 16).



I/O Signals

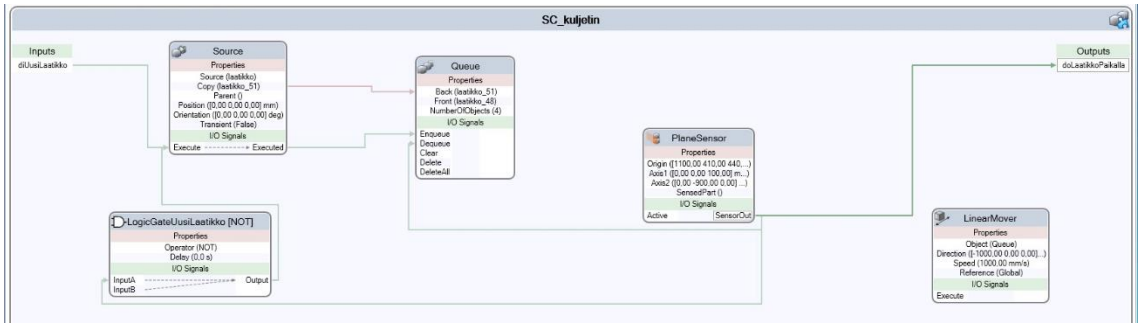
Name	Signal Type	Value
diUusiLaatikko	DigitalInput	0
doLaatikkoPaikalla	DigitalOutput	0

Add I/O Signals Expose Child Signal Edit Delete

I/O Connections

Source Object	Source Signal	Target Object	Target Signal
SC_kuljetin	diUusiLaatikko	Source	Execute
Source	Executed	Queue	Enqueue
PlaneSensor	SensorOut	Queue	Dequeue
PlaneSensor	SensorOut	SC_kuljetin	doLaatikkoPaikalla
PlaneSensor	SensorOut	LogicGateUusiLaatikko [NOT]	InputA
LogicGateUusiLaatikko [NOT]	Output	Source	Execute

Kuva 16. Kuljettimen I/O-signaalit.

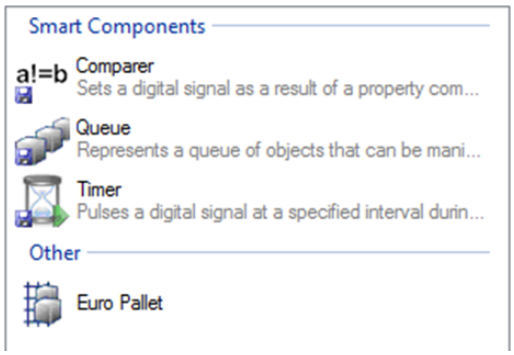


Kuva 17. Kuljettimen logiikkakaavio.

Kuljetin toimii seuraavasti: diUusiLaatikko-tulon ollessa aktiivinen lähettää se signaalin Source-komponentin tuloon Execute, joka luo uuden kopion laatikosta ja asettaa Executed-lähdön aktiiviseksi. Source-komponentin lähdöstä tuleva signaali asettaa Queue-komponentin Enqueue-tulon aktiiviseksi, jolloin luotu kopio liitetään jonon viimeiseksi jäseneksi. PlaneSensor-komponentin tunnistaessa kappaleen sen SensorOut-lähtö muuttuu aktiiviseksi ja lähettää signaalit Queue-komponentin Dequeue-tuloon, joka poistaa jonon ensimmäisen jäsenen jonosta, kuljettimen doKappalePaikalla-lähtöön, millä välitetään tieto robotin ohjaimelle, että kappale on noudettavissa sekä LogicGate(NOT)-komponentin InputA-tuloon, jolloin sen Output-lähtö ei ole aktiivinen ja koska ko. lähtö ei lähetä signaalia Source-komponentin Execute-tuloon ei Source-komponentti luo uutta laatikkoa vasta kun laatikko noudetaan ja PlaneSensor ei enää tunnista kappaletta.

3.3.3 Kuormalava

Kuormalavoista, joille työkappaleita lavataan, voidaan myös tehdä SC-komponentteja. Näin saadaan täydet lavat tyhjennettyä tai kuljetettua ulos solusta, jotta simulaatiota voidaan suorittaa useamman iteraatiokierroksen ajan (kuva 19).



Kuva 18. Kuormalavan komponentit.

Itsestään tyhjenevään kuormalavan toiminnot ja niihin tehdyt asetukset:

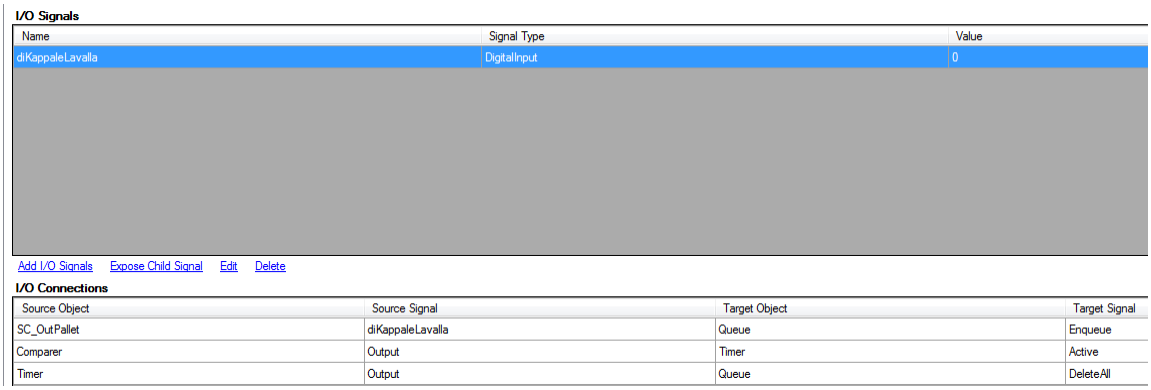
- Queue-jonokomponentti.

- Comparer-vertailijatoiminnon B-arvoksi valittiin työkappaliden määrä lavalla sen ollessa täynnä ja operaattoriksi valittiin yhtä kuin (==).
- Timer-komponentin intervalliarvoksi asetettiin 2s ja valittiin automaattinen toisto.

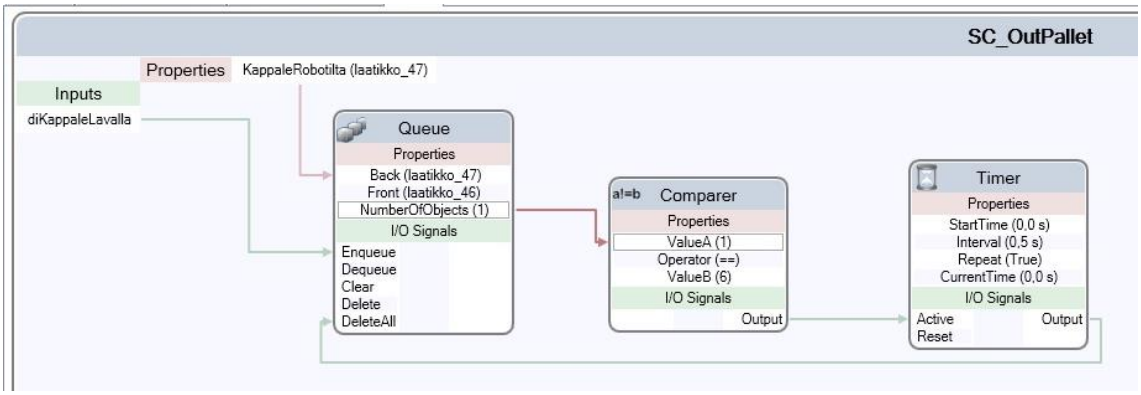
Properties and Bindings-välilehdellä luotiin dynaaminen ominaisuus, KappaleRobotilta, jolla saatiin välitettyä tarttujan tunnistaman kappaleen tiedot lavakomponentille. Tämä tieto sidottiin Queue-komponentin Back-ominaisuuteen, jotta tuotu kappale liitettiin aina joukon viimeiseksi. Queue-komponentin NumberOfObjects-tieto sidottiin Comparer-komponentin vertailtavaan A-arvoon (kuva 19).

Kuva 19. Kuormalavan sidokset.

Kuormalava-komponentille luotiin yksi digitaalinen tulo diKappaleLavalla (kuva 20).



Kuva 20. Kuormalavan I/O-signaalit.



Kuva 21. Kuormalavan logiikkakaavio.

Kuormalava toimii seuraavasti: kun diKappaleLavalla-tulo on aktiivinen lähettää se signaalin Queue-objektin Enqueue-tuloon, joka liittää tarttujalla tunnistetun kappaleen jonokomponentin viimeiseksi jäseneksi ja lisää NumberOfObjects-laskuriin yhden. Comparer-vertailija vertaa edellä mainittua arvoa vertailuarvoon ja kun ehto on tosi, komponentin Output-lähtö muuttuu aktiiviseksi ja lähettää Timer-

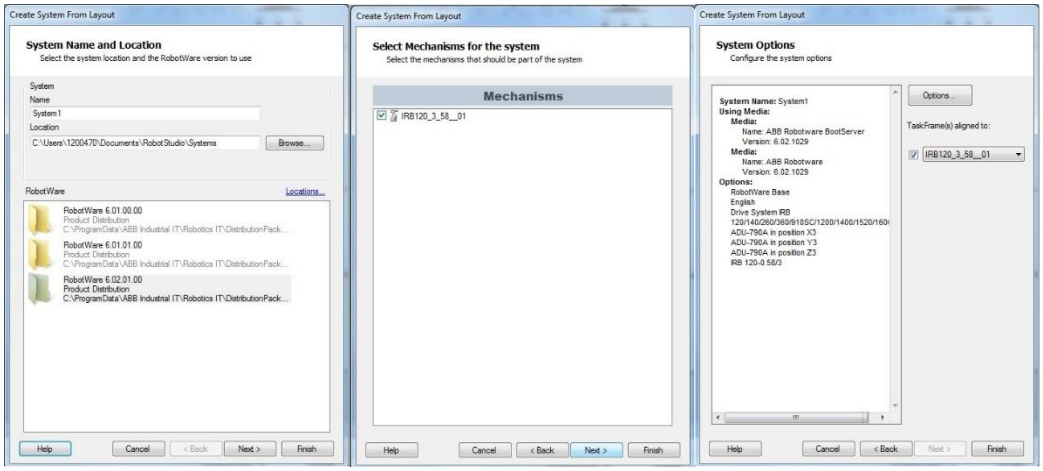
ajastinkomponentin Execute-tuloon signaalin ja ajastin käynnistyy. Ajastimeen asetetun ajan loputtua, komponentin Output-lähtö muuttuu aktiiviseksi ja lähettää jonokomponentin DeleteAll-tuloon signaalin, joka poistaa kaikki kappaleet jonossa, jolloin lava tyhjenee.

4 SOLUN ASETUKSET

Kun tarvittavat oheislaitteet on saatu luotua, voidaan aloittaa simuloidun robottisolun muokkaus. Helpointa on aloittaa tyhjästä simulaatiosta. Aluksi valitaan aloitusikkunassa Empty Station -painike.

4.1 Solun layout ja robotin ohjaimen lisääminen

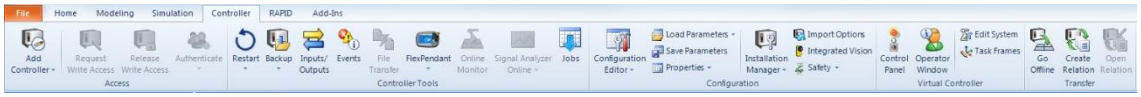
Luodut komponentit voidaan tuoda simulaatioon Import Library -toiminnolla, joka löytyy Home-välilehdeltä. Robotti voidaan tuoda ABB Library -toiminnolla, joka löytyy myös Home-välilehdeltä. Solulle luodaan layout siirtämällä komponentit haluttuihin asemiin. Kun solun layout on saatu muodostettua, voidaan robotille lisätä virtuaaliohjain. Valitaan Home-välilehdeltä Robot System -painike ja sen valikosta Add from Layout. Avautuu muokkausikkuna (kuva 22), missä annetaan virtuaaliohjaimelle nimi, tiedostopolku sekä ohjelmistoversio. Seuraavassa ikkunassa voidaan lisätä mekanismeja ohjaimelle, robotteja, lineaariratoja ym. Seuraavassa ikkunassa määritetään robotin peruskoordinaatiston nollapiste, Task Frame, joka on valmiiksi asetettu robotin sijaitsemaan koordinaattipisteeseen. Options-painikkeella voidaan ohjaimeen lisätä toimintoja. Finish-painikkeella virtuaaliohjain käynnistyy.



Kuva 22. Robotin ohjaimen asetusikkunat.

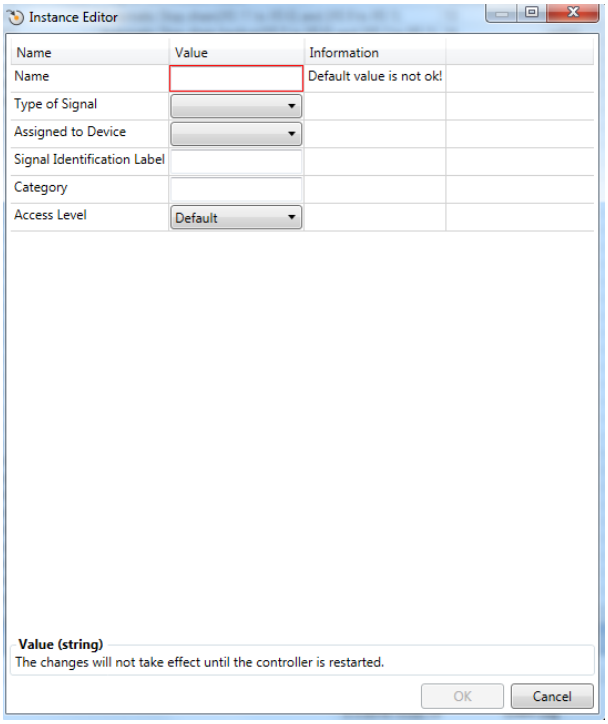
4.2 I/O-signaalit ja komponenttien linkitys

Solun asettelun ja ohjaimen lisäämisen jälkeen, on ohjaimeen lisättävä simulaatiossa vaadittavat I/O-signaalit. Controller-välilehdeltä (kuva 23) valitaan Controller Configuration -painike, jonka valikosta valitaan I/O-System.

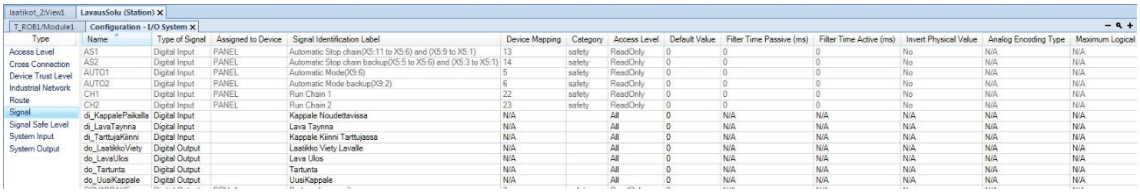


Kuva 23. Controller-välilehti.

Virtuaaliohjaimen signaali-ikkuna avautuu ja valitaan siitä kohta Signal. Painetaan hiiren oikeaa painiketta Signal-otsikon päällä ja valitaan New Signal. Muokkausikkunassa (kuva 24) määritetään signaalin tyyppi, signaalin nimi, selite sekä käyttöoikeus.



Kuva 24. Uuden I/O-signaalin muokkausikkuna.



Kuva 25. Laatikkosoluun luodut I/O-signaalit.

Signaalien luomisen jälkeen ohjain on käynnistettävä uudelleen, jotta luodut signaalit ovat käytettävissä. Valitaan Controller-välilehdeltä Restart-painike. Ohjaimen käynnistyttyä uudelleen, on virtuaaliohjain ja luodut Smart Component-laitteet linkitettävä toisiinsa Simulation-välilehdeltä (kuva 26) löytyvän Station Logic -toiminnon avulla.

Kuva 26. Simulation-välilehti.

Painiketta painettaessa avautuu muokkausikkuna, joka muistuttaa Smart Componentien luonnissa käytettyä ikkunaa. Compose-välilehdellä voidaan simulaatioon lisätä yksittäisiä Smart Component -objekteja. Bindings-välilehdellä voidaan oheislaite SC-komponentteihin tehdyt dynaamiset ominaisuudet sitoa toisiinsa. Signals and Connections-välilehdellä luodaan komponenttien ja virtuaaliohjaimen väliset I/O-yhteydet.

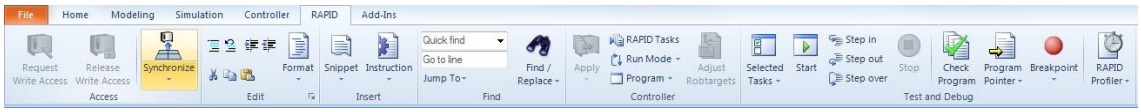
Design välilehdellä voidaan tarkastella komponenttien ja ohjaimen välisiä yhteyksiä logiikkakaaviona.

4.3 Liikeratojen luominen ja RAPID-muokkaus

Komponenttien linkityksen jälkeen voidaan robottisolulle alkaa luomaan ohjelmaa, jota suoritetaan simulaatiossa. Home-välilehdeltä löytyvien Freehand-liikutustyökalujen avulla voidaan robottia liikuttaa ja Path Programming -työkalujen avulla voidaan tallentaa koordinaattipisteet ohjaimelle. Ruudun alareunassa on valikko, missä voidaan valita liiketyyppi, - nopeus ja zone-arvo pisteille. Luodut koordinaattipisteet löytyvät vasemmassa reunassa olevassa ikkunassa valitsemalla Paths&Targets-välilehden. Välilehdellä voi muokata luotuja pisteitä, robotin konfiguraatioita pisteissä ym.

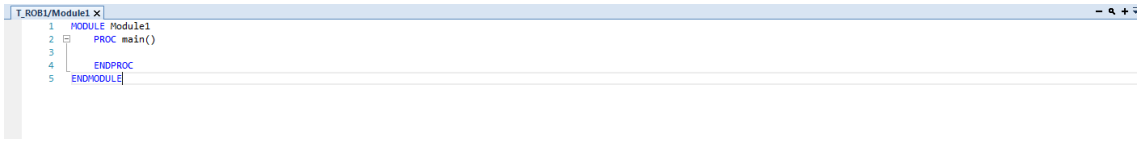
Samasta valikosta löytyy Paths-kansio mihin voidaan luoda uusia liikeratoja opetetuista koordinaattipisteistä. Uusi rata voidaan lisätä joko valitsemalla Home-välilehdeltä Paths-painike tai painaa hiiren oikeaa painiketta vasemman reunan ik-

kunan Paths-kohdassa ja valita Create new Path. Luodut pisteet voidaan nyt siirtää luotuun Path- objektiin. Liikerata voidaan tarkastaa valitsemalla Move Along Path-toiminto, jolloin robotti liikkuu luotua rataa pitkin. Rataan voidaan lisätä myös toimintokäskyjä painamalla hiiren oikeaa painiketta radan nimen päällä ja valitsemalla Insert Action Instruction. Nämä käskyt ovat toisaalta helpompi lisätä ohjelman tekstieditorissa. Luotu rata ja koordinaattipisteet voidaan synkronisoida robotin ohjaimelle toiminnolla Synchronize to RAPID-toiminnolla. RAPID-välilehdeltä (kuva 27) voidaan nyt tarkastella luotua ohjelmaa valitsemalla Program Pointer -> Go To Program Pointer.



Kuva 27. RAPID-välilehti.

RAPID-editorissa (kuva 28) näkyvät luodut koordinaattipisteet yläreunassa ja robotin ohjelma niiden alapuolella. Editorissa voidaan muokata robotin ohjelmaa esimerkiksi lisäämällä I/O-käskyjä tai lisäämällä ohjelman rakenteeseen silmukoita.



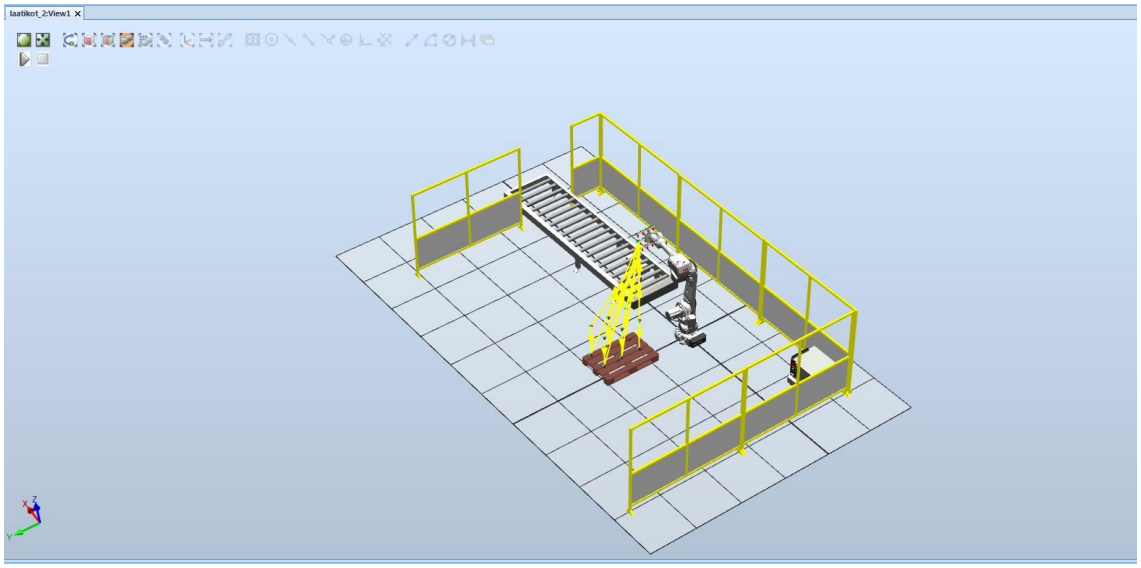
Kuva 28. Tyhjä RAPID-editori.

5 LUODUT SIMULAATIOT

5.1 Laatikkosolu

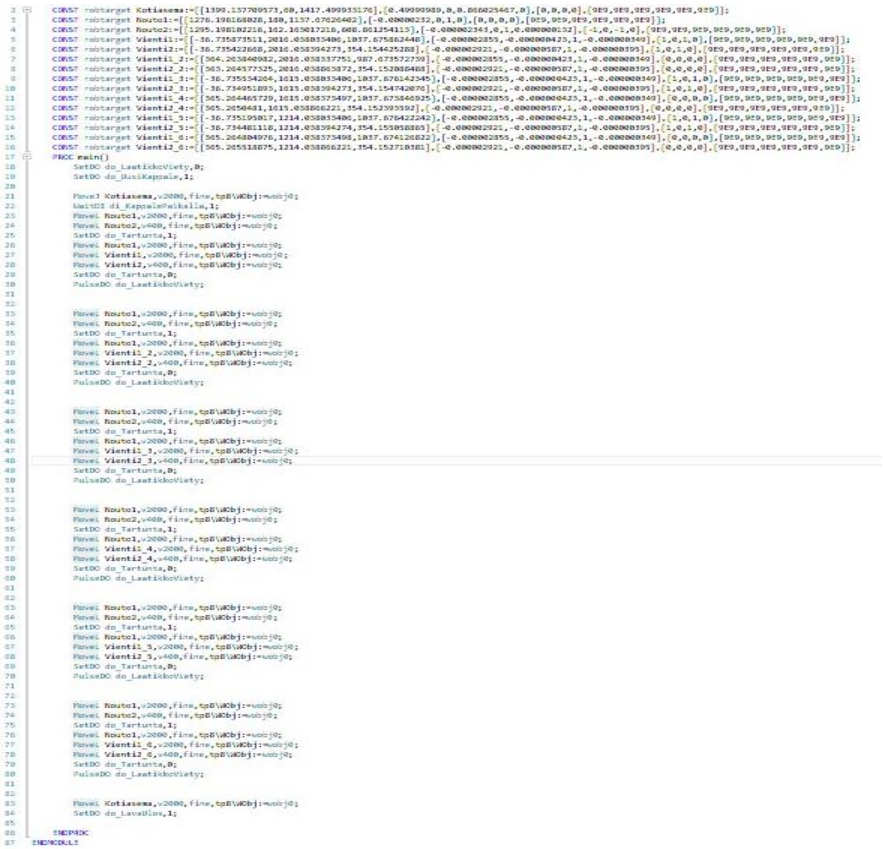
Työn tarkoituksena oli luoda kolme simuloitua robottisolua, joita voitaisiin käyttää robotiikan opetuksessa ABB robottien, etäohjelmoinnin ja robottisolujen simuloinnin perehdytykseen.

Kuvassa 29. laatikkosolu-simulaation layout.



Kuva 29. Laatikkosolu.

Laatikkosolu on luoduista simulaatioista yksinkertaisin. Solussa on kuljetin, joka tuo laatikkoja robotille. Kuljettimen päässä on valoverho joka tunnistaa laatikon ja pysäyttää kuljettimen, kunnes laatikko on noudettu. Robotti noutaa laatikoita niiden saapuessa noutopaikalle ja lavaa laatikot kuormalavalle. Kuormalavan täyttyessä lava tyhjenee itsestään ja ohjelman kierto alkaa alusta (kuva 30).



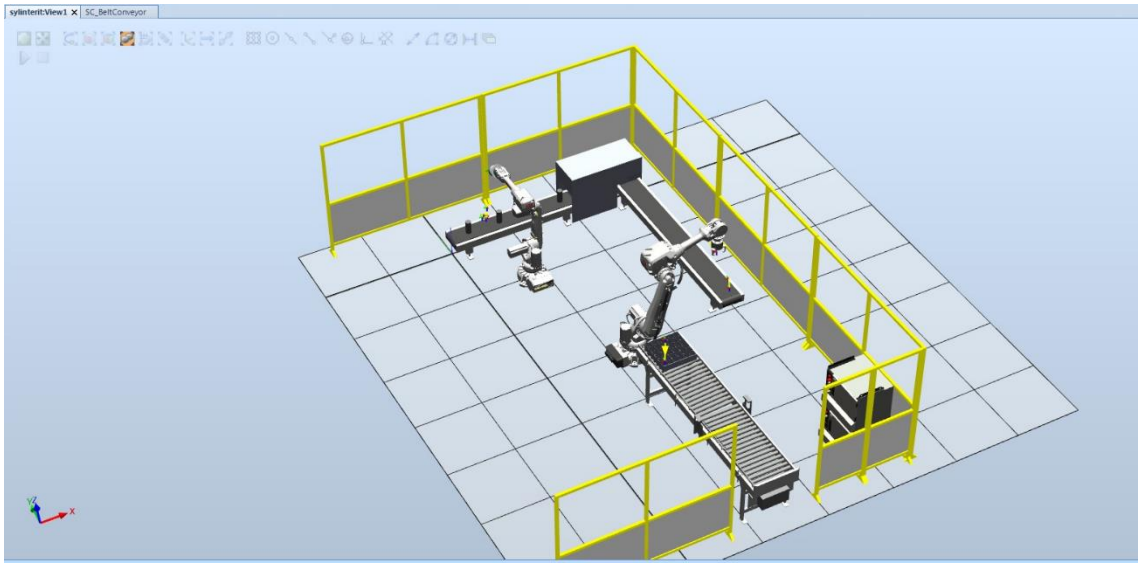
```
CONST robtarget Kotiasema:=[1399.157709573,69,1417.499935176],[0.49999989,0,0.866025467,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Nouto1:=[1276.196168028,188,1157.67626482],[-0.00000232,0,1,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Nouto2:=[1295.198182216,162.163817218,608.881254115],[-0.000002343,0,1,0.000000132],[-1,0,-1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti1:=[-36.775877511,2016.058835486,1837.675862448],[-0.000002855,-0.000000425,1,-0.000000349],[1,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti2:=[-36.775422668,2016.058394273,354.154425288],[-0.000002921,-0.000000587,1,-0.000000395],[1,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti_2:=[364.265844982,2016.058537751,987.673572739],[-0.000002855,-0.000000425,1,-0.000000349],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti2_2:=[365.264577325,2016.058863272,354.153866468],[-0.000002921,-0.000000587,1,-0.000000395],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti_3:=[-36.735554264,1815.058835486,1837.676142545],[-0.000002855,-0.000000425,1,-0.000000349],[1,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti2_3:=[-36.734951895,1815.058394273,354.154742076],[-0.000002921,-0.000000587,1,-0.000000395],[1,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti_4:=[365.264465729,1815.058375497,1837.675846925],[-0.000002855,-0.000000425,1,-0.000000349],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti2_4:=[365.2650481,1815.058866221,354.152395592],[-0.000002921,-0.000000587,1,-0.000000395],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti_5:=[-36.735195817,1214.058835486,1837.676422242],[-0.000002855,-0.000000425,1,-0.000000349],[1,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti2_5:=[-36.734481118,1214.058394274,354.153858865],[-0.000002921,-0.000000587,1,-0.000000395],[1,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti_6:=[365.264894976,1214.058375498,1837.674128812],[-0.000002855,-0.000000425,1,-0.000000349],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Vienti2_6:=[365.265118675,1214.058866221,354.152718181],[-0.000002921,-0.000000587,1,-0.000000395],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
PROC main()
    SetDO do_LaatikkoViety,0;
    SetDO do_UusiKappale,1;

    MoveJ Kotiasema,v2000,fine,tpB\WObj:=wobj0;
    WaitDI di_KappaleMaikalle,1;
    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Nouto2,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,1;
    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti2,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,0;
    PulseDO do_LaatikkoViety;


    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Nouto2,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,1;
    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti_2,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti2_2,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,0;
    PulseDO do_LaatikkoViety;


    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Nouto2,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,1;
    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti_3,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti2_3,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,0;
    PulseDO do_LaatikkoViety;


    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Nouto2,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,1;
    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti_4,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti2_4,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,0;
    PulseDO do_LaatikkoViety;


    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Nouto2,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,1;
    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti_5,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti2_5,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,0;
    PulseDO do_LaatikkoViety;


    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Nouto2,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,1;
    MoveL Nouto1,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti_6,v2000,fine,tpB\WObj:=wobj0;
    MoveL Vienti2_6,v400,fine,tpB\WObj:=wobj0;
    SetDO do_Tartunta,0;
    PulseDO do_LaatikkoViety;


    MoveL Kotiasema,v2000,fine,tpB\WObj:=wobj0;
    SetDO do_LavaDies,1;

ENDPROC
ENDMODULE
```

Kuva 30. Laatikkosolussa käytetty ohjelma.

Ohjelmaeditorin yläreunassa näkyvät robotille opetetut koordinaattipisteet, jotka ovat ohjelmassa vakioita. Seuraavaksi ohjelma asettaa kuljettimen do_UusiKappale-tulon aktiiviseksi, jolloin kuljetinkomponentti luo uuden laatikon ja kuljettaa sen noutopaikalle. Robotti liikkuu kotiasemaan ja odottaa, että laatikko on noudettavissa. Seuraavaksi se liikkuu hakemaa laatikon ja vie sen ensimmäiseen opetettuun lastauspisteeseen. Kun laatikko on viety lavalle, lähettää ohjain lavan SC-komponentille pulssin, että laatikko on lavalla ja SC-komponentin laskuri lisää arvoonsa yhden. Seuraavaksi robotti noutaa uuden laatikon ja tuo sen lavalle, toiseen opetettuun lastauspisteeseen. Kaikki lastauspisteet on opetettu robotille erikseen. Tätä toistetaan, kunnes lava on täynnä.

5.2 Sylinterisolu

Kuvassa 31. sylinterisolu-simulaation layout.



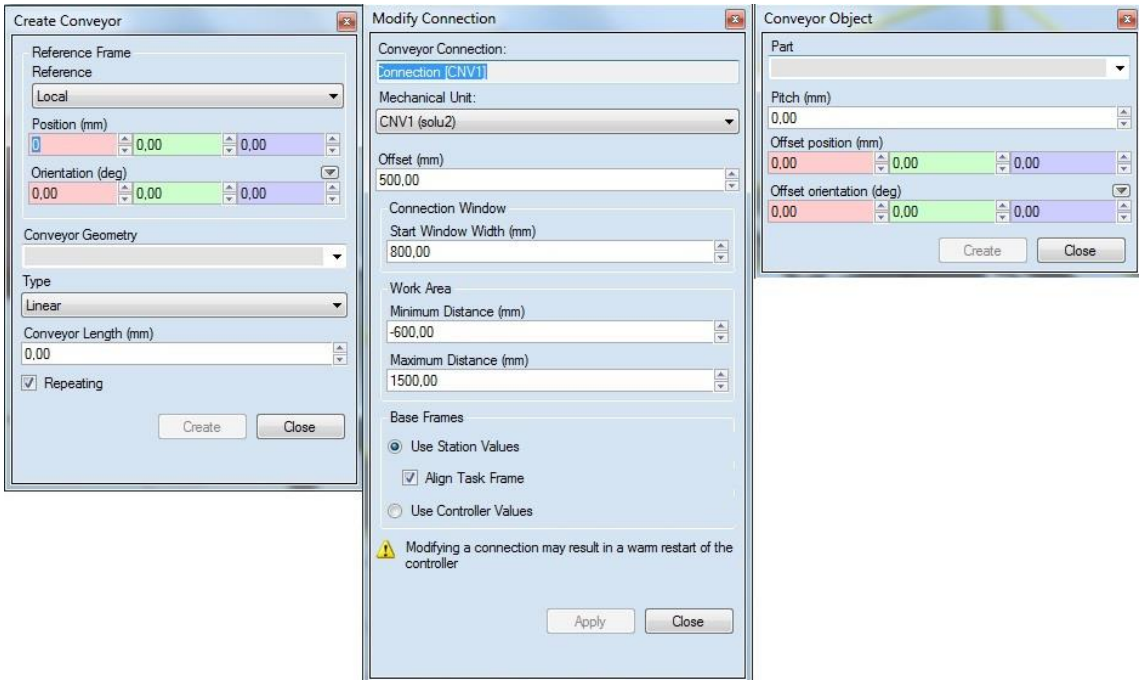
Kuva 31. Sylinterisolu.

Sylinterisolu on jo aikaisempaa laatikkosolua huomattavasti monimutkaisempi. Solussa ensimmäiseltä kuljettimelta saapuu soluun sylintereitä, joille ensimmäinen robotti suorittaa hionnan sylintereiden yläreunaan. Kuljetin ei pysähdy hionnan ajaksi vaan robotti hioo kappaleen sen liikkuessa kuljettimella. Tässä käytettiin Conveyor Tracking -toimintoa. Hionnan jälkeen kuljetin siirtää kappaleet välivarastoon, mistä niitä syötetään yksitellen toiselle kuljettimelle. Toisen kuljettimen päädyssä on sensori joka tunnistaa kappaleen ja pysäyttää kuljettimen, kunnes kappale on noudettu. Robotti lavaa sylinterit kolmannella kuljettimella olevalle 6x6 paikkaiselle tarjottimelle. Tarjottimen tullessa täyteen, tyhjenee se itsestään ja simulaation kierto alkaa alusta (kuva 33 ja 34).

Conveyor Tracking eli kuljetinseuranta on toiminto ABB:n robottiohjaimissa, millä voidaan työstää työkappaleita kuljettimella pysäyttämättä kuljetinta. Conveyor Tracking luodaan seuraavasti. Kun lisätään soluun robotin ohjain, valitaan Options-painike. Valitaan valikosta Motion Coordination-> Conveyor Tracking, jolloin

aukeaa ikkuna Select dependency chain, valitaan ylempi vaihtoehto. Seuranta on nyt lisätty ja ohjain voidaan käynnistää.

Seuraavaksi luodaan kuljetin, Create Conveyor-toiminnolla Modeling-välilehdeltä. Vasemmalle layout-ikkunan yläpuolelle aukeaa kuljetin-muokkausikkuna (kuva 32). Valitaan kuljettimen geometriamalli, sen koordinaattipiste, kuljettimen tyyppi sekä kuljettimen pituus. Seuraavaksi luodaan työskentelyalue kuljettimelle valitsemalla luotu kuljetinobjekti layout-ikkunasta ja painetaan hiiren oikeaa painiketta sen päällä. Avautuvasta ikkunasta asetetaan työalue ja rastitetaan valinta Align Task Frame. Seuraavaksi tuodaan tai mallinnetaan työkappale ja lisätään se kuljettimen objektiksi. Painetaan pientä nuolta kuljetinobjektin vieressä layout-ikkunassa ja valitaan hiiren oikealla Object Source ja siitä Add Object. Avautuvassa ikkunassa valitaan työkappaleen malli ja Pitch, eli kuinka suuren välimatkan päässä toisistaan kappaleet kuljettimella ovat. Painetaan nuolta Object Source-kohdan vieressä ja valitaan sen alla oleva työkappale-objekti hiiren oikealla painikkeella. Seuraavaksi valitaan Place on Conveyor, jolloin kappale ilmestyy kuljettimen päätyyn. Valitaan seuraavaksi Attach WorkObject -valinnasta ainoa valinta mikä on saatavissa. Kappale voidaan nyt Jog-toiminnolla liikuttaa työalueelle ja luoda liikeratoja työstämistä varten.



Kuva 32. Create conveyor- ikkunat.

```
MODULE Module1
    CONST robtarget Target_10:=[[90.5,-0.717,650],[0.022518268,0.706948689,-0.706748136,0.014951646],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,0]];
    CONST robtarget Target_20:=[[90.5,-0.717,500],[0.022518268,0.706948689,-0.706748136,0.014951646],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,0]];
    CONST robtarget Target_30:=[[50,39.783,500],[0.022518268,0.706948689,-0.706748136,0.014951646],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,0]];
    CONST robtarget Target_40:=[[9.5,-0.717,500],[0.022518268,0.706948689,-0.706748136,0.014951646],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,0]];
    CONST robtarget Target_50:=[[50,-41.217,500],[0.022518268,0.706948689,-0.706748136,0.014951646],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,0]];
    CONST robtarget Target_60:=[[90.5,-0.717,500],[0.022518268,0.706948689,-0.706748136,0.014951646],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,0]];
    CONST robtarget Target_70:=[[90.5,-0.717,650],[0.022518268,0.706948689,-0.706748136,0.014951646],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,0]];
    CONST jointtarget Koti:=[[0,0,0,0,30,0],[9E9,9E9,9E9,9E9,9E9,0]];
    PROC Path_10()
        MoveL Target_10,v1000,z100,Jaysteenpoistaja\WObj:=wobj_cnv1;
        MoveL Target_20,v1000,z100,Jaysteenpoistaja\WObj:=wobj_cnv1;
        MoveC Target_30,Target_40,v1000,z100,Jaysteenpoistaja\WObj:=wobj_cnv1;
        MoveC Target_50,Target_60,v1000,z100,Jaysteenpoistaja\WObj:=wobj_cnv1;
        MoveL Target_70,v1000,z100,Jaysteenpoistaja\WObj:=wobj_cnv1;
    ENDPROC
    PROC main2()
        MoveAbsJ Koti,v1000,fine,Jaysteenpoistaja\WObj:=wobj0;
        ActUnit CNV1;
        WaitWObj wobj_cnv1;
        Path_10;
        MoveAbsJ Koti,v1000,fine,Jaysteenpoistaja\WObj:=wobj0;
        DropWObj wobj_cnv1;
    ENDPROC
ENDMODULE
```

Kuva 33. Sylinterin hionnassa käytetty ohjelma.

Kuvan yläreunassa näkyvät luodut koordinaattipisteet. Pääohjelma main2() liikuttaa robotin ensin kotiasemaan, jonka jälkeen se aktivoi sylintereitä liikuttavan kuljettimen. Seuraavaksi robotti odottaa, että sylinteri saapuu työalueelle ja kutsuu aliohjelma Path_10(), joka on robotin liikerata hionnassa. Sylinteristä on luotu

WorkObject-työkappale robotin muistiin, jolloin luotu liikerata liikkuu kappaleen mukana ja robotti suorittaa liikkeet kappaleen liikkuessa. Kun aliohjelma on suoritettu, liikkuu robotti takaisin kotiasemaan ja pudottaa työstämänsä kappaleen muistista. Robotti toistaa tätä ohjelmaa aina kun se tunnistaa sylinterin työalueella.

```
7     VAR num laskuri1;
8
9     PROC main()
10        MoveJ Kotiasema,v5000,fine,Vakiotarttuja_1\WObj:=wobj0;
11        SetDO do_Kiinnita,0;
12        laskuri1:=0;
13        WHILE laskuri1<36 DO
14        PulseDO do_UusiKappale;
15        WaitDI di_KappalePaikalla,1;
16        nouto;
17        MoveJ Kotiasema,v5000,fine,Vakiotarttuja_1\WObj:=wobj0;
18        offsetVienti;
19        MoveJ Kotiasema,v5000,fine,Vakiotarttuja_1\WObj:=wobj0;
20        ENDWHILE
21    ENDPROC
22
23    PROC nouto()
24        MoveL Nouto1,v3000,fine,Vakiotarttuja_1\WObj:=wobj0;
25        MoveL Nouto2,v400,fine,Vakiotarttuja_1\WObj:=wobj0;
26        SetDO do_Kiinnita,1;
27        WaitTime 0.7;
28    ENDPROC
29
30    PROC offsetVienti()
31        MoveL offs (Vienti1,(laskuri1 DIV 6)*83,(laskuri1 MOD 6)*83,0),v3000,fine,Vakiotarttuja_1\WObj:=wobj0;
32        MoveL offs (Vienti2,(laskuri1 DIV 6)*83,(laskuri1 MOD 6)*83,0),v500,fine,Vakiotarttuja_1\WObj:=wobj0;
33        SetDO do_Kiinnita,0;
34        WaitTime 0.7;
35        PulseDO do_KappaleViety;
36        MoveL offs (Vienti1,(laskuri1 DIV 6)*83,(laskuri1 MOD 6)*83,0),v3000,fine,Vakiotarttuja_1\WObj:=wobj0;
37        laskuri1:=laskuri1+1;
38    ENDPROC
39 ENDMODULE
```

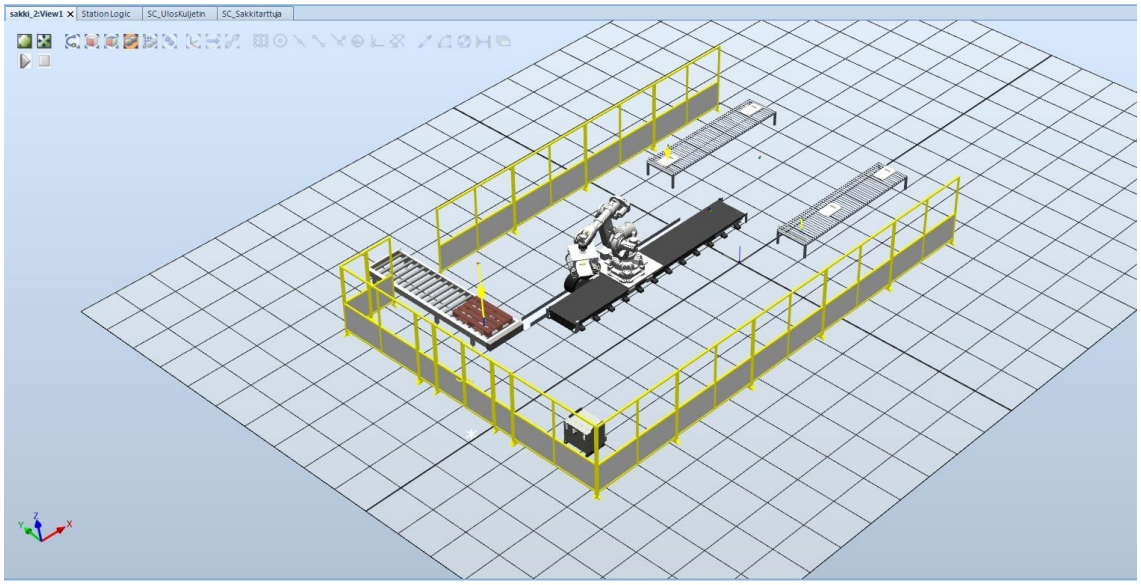
Kuva 34. Sylintereiden lavaukseen käytetty ohjelma.

Pääohjelma liikuttaa robotin ensin kotiasemaan ja asettaa tarttujan auki. Seuraavaksi asetetaan luodun numeerisen muuttujan laskuri1- arvo nollaksi. Seuraavaksi on While-silmukka, jonka sisällä olevaa ohjelmakoodia toistetaan, kunnes laskuri1-muuttujan arvo on 36. Silmukan sisältämässä ohjelmassa käsketään kuljettimen luoda uusi kappale, jonka jälkeen robotti odottaa, että kappale on noudettavissa ja kutsutaan nouto- aliohjelma. Aliohjelman suorituksen jälkeen robotti liikkuu kotiasemaan ja kutsuu offsetVienti-aliohjelman, jonka suoritettua liikkuu se taas kotiasemaan.

Nouto aliohjelmassa on opetettu robotille kaksi koordinaattipistettä, sekä I/O-käsky tarttujan sulkemiseen. Vienti ohjelmassa on myös opetettu koordinaattipisteet kappaleen viemiseksi lavalle, mutta koska kappaletta ei viedä aina samaan pisteeseen on pisteiden siirrossa käytetty offset-toimintoa. Pisteiden vaakasuuntaiseen siirron laskemiseen on käytetty kertoimena laskuri1-muuttujan arvon sekä vaakasuuntaisten tarjotinpaikkojen jakojäännöstä, MOD-funktiota. Pystysuuntaisen siirron kertoimena on käytetty samojen arvojen kokonaisluku jakolaskua DIV-funktiota. Kun kappale on viety lavalle, lähettää ohjain pulssin tarjottimen SC-komponentille. Kappaleen viennin jälkeen laskuri1-muuttujaan lisätään 1.

5.3 Säkkisolu

Kuvassa 35. Säkkisolu-simulaation layout.



Kuva 35. Säkkisolu.

Säkkisolussa on kaksi kuljetinta jotka kuljettavat säkkejä soluun. Robotti käy vuorotellen noutamassa säkin kuljettimelta ja liikkuu lineaariradalla kolmannelle kuljettimelle, missä se lavaa säkit kuljettimella olevalle kuormalavalle. Kuormalavan

täyttyessä, kolmas kuljetin siirtää täyden lavan ulos solusta ja luo uuden tyhjän lavan edellisen tilalle.

```
11      VAR num laskuri;
12      VAR num laskuri2;
13      VAR num laskuri3;
14
15      PROC main()
16          ActUnit TRACK_1;
17          SetDO do_UusiLava,0;
18          SetDO do_Kiinnita,0;
19          SetDO do_LavaTaynna,0;
20          SetDO do_UusiSakki,1;
21          PulseDO do_UusiLava;
22          laskuri:=0;
23          laskuri2:=0;
24          laskuri3:=0;
25          MoveAbsJ Kotiasema,v7000,z200,Sakkitarttuja_1\WObj:=wobj0;
26          WaitDI di_SakkiPaikalla,1;
27          WHILE laskuri<12 DO
28              IF laskuri MOD 2=0 THEN
29              Nouto1;
30              MoveAbsJ Kotiasema,v7000,z200,Sakkitarttuja_1\WObj:=wobj0;
31              ELSE
32              Nouto2;
33              MoveAbsJ Kotiasema,v7000,z200,Sakkitarttuja_1\WObj:=wobj0;
34              ENDIF
35              IF laskuri<4 THEN
36              Kerros_1;
37              MoveAbsJ Kotiasema,v7000,z200,Sakkitarttuja_1\WObj:=wobj0;
38              ELSEIF laskuri2<4 THEN
39              Kerros_2;
40              MoveAbsJ Kotiasema,v7000,z200,Sakkitarttuja_1\WObj:=wobj0;
41              ELSE
42              Kerros_3;
43              MoveAbsJ Kotiasema,v7000,z200,Sakkitarttuja_1\WObj:=wobj0;
44              ENDIF
45          ENDWHILE
46          SetDO do_LavaTaynna,1;
47          WaitTime 5;
48      ENDPROC
```

Kuva 36. Säkkisolun pääohjelma.

Pääohjelmassa ensin asetetaan lineaarirata aktiiviseksi ja varmistetaan ettei lähtöjä ole jäänyt aktiivisiksi. Tämän jälkeen ohjelma luo uuden lavan sekä säkit. Laskuri-muuttujat nollataan. Seuraavaksi robotti odottaa säkin olevan noudettavissa. While-silmukkaa suoritetaan, niin kauan kuin laskuri- muuttuja on alle 12. Mikäli laskuri- muuttujan ja luvun 2 jakojäännös on nolla, suoritetaan Nouto1-

aliohjelma ja mikäli se ei ole nolla suoritetaan Nouto2. Seuraavaksi IF-ehto, mikäli laskuri-muuttuja on alle 4, suoritetaan Kerros_1-aliohjelmaa. Jos laskuri-muuttuja on 4 tai yli ja laskuri2-muuttuja on alle 4 suoritetaan Kerros_2-aliohjelma. Jos laskuri- ja laskuri2-muuttujat ovat kummatkin 4 tai yli suoritetaan Kerros_3-aliohjelma. Kun laskuri-muuttuja on arvossa 12, loppuu while-silmukan suoritus, jolloin lava on täynnä. Viimeiselle kuljettimelle tulee signaali, että lava on täynnä, jolloin se kuljettaa täyden lavan solusta ulos ja luo tyhjän lavan tilalle.

6 POHDINTA

Työssä luotiin kolme simuloitua robottisolua RobotStudio-ohjelmistolla. Simulaatioita on tarkoitus käyttää myöhemmin robotiikan opetuksessa Turun ammattikorkeakoulussa. Soluja luodessa oli tarkoitus samanaikaisesti opetella ja tutustua ohjelmiston toimintoihin ja käyttöön.

Projektin aihetta rajatessa asetettiin tavoitteeksi luoda kolme erilaista virtuaalista robottisolua, jotka käsittelivät kappaleenkäsittelyä. Työkappaleiksi valittiin laatikko, sylinterin mallinen kappale sekä säkki. Tämä siitä syystä, että kappaleisiin tarttuminen robotilla olisi mahdollisimman erilaista. Soluihin oli tavoitteena saada muutenkin mahdollisimman paljon vaihtelua muun muassa erilaisilla ohjelmointitavoilla ja oheislaitteilla. Pääasialliset tavoitteet saavutettiin loistavasti ja toissijaiseksi asetettu tavoite soluilla tehtävien harjoitusten suunnittelusta toteutui myös kiitettävästi. Esimerkiksi laatikkosolu-simulaatioon suunniteltu harjoitus, jossa robotin ohjelma pitäisi tehdä järkevämmällä tavalla.

Työn aikana oppi paljon robottisolun suunnittelusta ja siinä esiintyvistä haasteista sekä tuotannon simulaatioiden tekemisestä ja simulaatioissa vastaan tulevista ongelmista. Opittua tietoa on varmasti mahdollista hyödyntää teolliseen tuotantoon suuntaavalla alalla. Työn aikana sai myös hyvää oppia muun muassa projektin aikatauluttamisesta ja projektin läpiviemisestä, josta on ehdottomasti myös hyötyä tulevaisuuden työuralla.

LÄHTEET

Aalto H., Heilala J., Hirvelä T., Kuivanen R., Laitinen M., Lehtinen H., Lempiäinen J., Lylynoja A., Renfors J., Selin K., Siintoharju T., Temmes J., Tuovila T., Veikkolainen M., Vihinen J., Virtanen A. 1999. Robotiikka. Vantaa: Talentum Oyj/Metallitekniikka

ABB. 2015. Operating manual, RobotStudio 6.02. https://library.e.abb.com/public/3e67eb2f82e44002a2e2fc4bfa0ff8d0/3HAC032104-en.pdf

Anandan T.M. 2015. Robotics 2015 and beyond: Collaboration, Connectivity, Convergence. Web-julkaisu. Viitattu 18.1.2016. http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Insights/Robotics-2015-and-Beyond-Collaboration-Connectivity-Convergence/content_id/5188

IFR. 2015. World robotics 2015. Viitattu 6.1.2016. http://www.ifr.org/industrial-robots/statistics/

Keinänen T., Kärkkäinen P., Lähetkangas M., Sumujärvi M., 2007. Automaatiojärjestelmien logiikat ja ohjausjärjestelmät. Helsinki: WSOY Oppimateriaalit OY

Nocks L. 2008. The Robot: The Life Story of a Technology. USA: Greenwood Publishing Group Inc.

Sclater N., Chironis N., 2006. Mechanisms and Mechanical devices. 4. painos. USA: McGraw-Hill